

2. Accessing Series and DataFrames

1. Access a single column(Series)

Use `DataFrameName['ColumnName']` syntax to access a Series.

```
main.py > ...
3  data = {
4      'A' : [1, 2, 3, 4],
5      'B' : [5, 6, 7, 8],
6      'C' : ['foo', 'bar', 'baz', 'sas']
7  }
8
9  df = pd.DataFrame(data)
10 print(df['A'])
```

PROBLEMS OUTPUT TERMINAL ... Python

```
PS C:\Users\ranga\Desktop\python> python main.py
0    1
1    2
2    3
3    4
Name: A, dtype: int64
PS C:\Users\ranga\Desktop\python> 
```

2. Access a multiple columns(Series) at once

To access multiple column, you need to provide a list column names into the above syntax.

Ex: If you want to access A, C columns the list of the columns names will be `['A', 'C']`.

Then use the same syntax, `DataFrameName[listOfColumnNames]`

`df[['A', 'C']]`

```
main.py > ...
3  data = {
4      'A' : [1, 2, 3, 4],
5      'B' : [5, 6, 7, 8],
6      'C' : ['foo', 'bar', 'baz', 'sas']
7  }
8  df = pd.DataFrame(data)
9  print(df[['A', 'C']])
```

PROBLEMS OUTPUT TERMINAL ... Python

```
PS C:\Users\ranga\Desktop\python> python main.py
   A  C
0  1  foo
1  2  bar
2  3  baz
3  4  sas
PS C:\Users\ranga\Desktop\python> 
```

3. loc method

loc method is **label-based**, which means it's used to select data by row and column labels(names/strings)

Syntax:

```
DataFrameName.loc[rowLabel, columnLabel]
```

```
main.py > ...
1  import pandas as pd
2  data = {
3      'Player': ['Alice', 'Bob', 'Charlie'],
4      'Score': [85, 90, 95],
5      'Matches': [10, 15, 20]
6  }
7  df = pd.DataFrame(data)
8  df.set_index('Player', inplace=True)
```

1. Selecting a single row by label

```
df.loc['Alice']
```

```
10  print(df.loc['Alice'])
```

PROBLEMS OUTPUT TERMINAL ... Python

```
PS C:\Users\ranga\Desktop\python> python main.py
Score      85
Matches    10
Name: Alice, dtype: int64
PS C:\Users\ranga\Desktop\python>
```

2. Selecting multiple rows by labels

Provide the list of the row labels to the loc method.

```
df.loc[['Alice', 'Bob']]
```

```
10  print(df.loc[['Alice', 'Bob']])
```

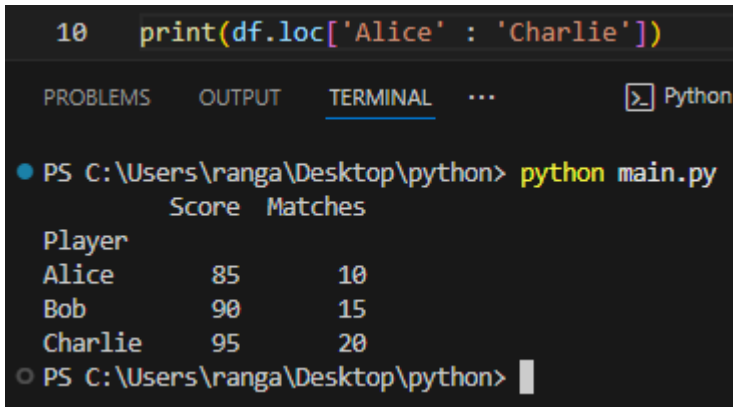
PROBLEMS OUTPUT TERMINAL ... Python

```
PS C:\Users\ranga\Desktop\python> python main.py
Score  Matches
Player
Alice    85     10
Bob     90     15
PS C:\Users\ranga\Desktop\python>
```

3. Selecting a range of rows by labels

```
df.loc['Alice':'Charlie']
```

```
10 print(df.loc['Alice' : 'Charlie'])
```



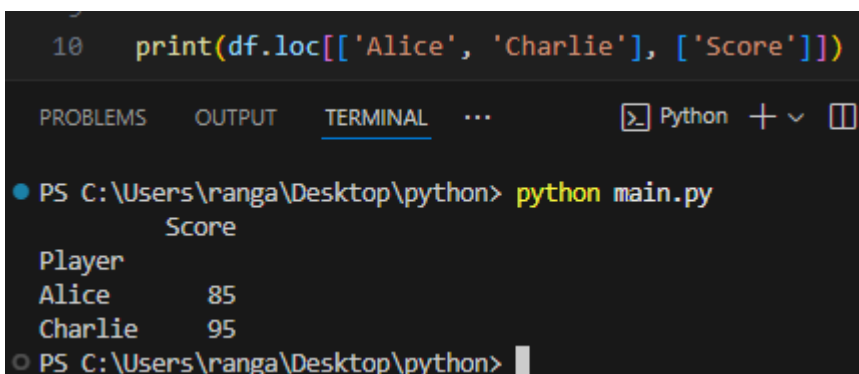
| Player | Score | Matches |
|---------|-------|---------|
| Alice | 85 | 10 |
| Bob | 90 | 15 |
| Charlie | 95 | 20 |

4. Selecting specific rows and columns by labels

Ex: Print Alice and Charlie's Scores.

```
df.loc[['Alice', 'Charlie'], ['Score']]
```

```
10 print(df.loc[['Alice', 'Charlie'], ['Score']])
```



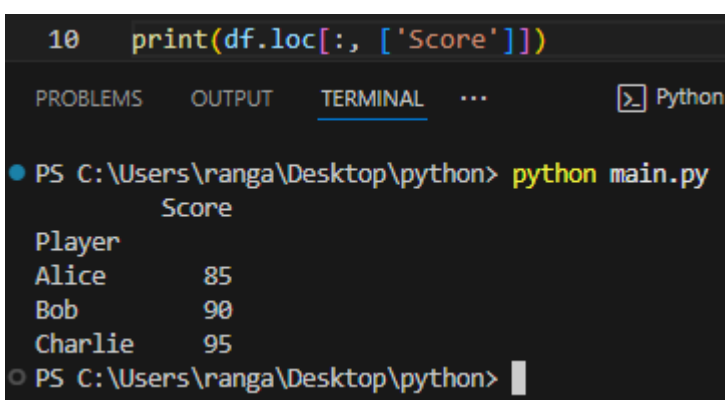
| Player | Score |
|---------|-------|
| Alice | 85 |
| Charlie | 95 |

5. Selecting all rows and specific columns

Ex: Print all player's scores.

```
df.loc[:, ['Score']]
```

```
10 print(df.loc[:, ['Score']])
```



| Player | Score |
|---------|-------|
| Alice | 85 |
| Bob | 90 |
| Charlie | 95 |

4. `iloc` method

`iloc` method is **integer position-based**, which means it's used to select data by row and column indices.

Syntax:

```
DataFrameName.iloc[rowIndex, columnIndex]
```

```
main.py > ...
1  import pandas as pd
2  data = {
3      'Player': ['Alice', 'Bob', 'Charlie'],
4      'Score': [85, 90, 95],
5      'Match': [10, 15, 20]
6  }
7  df = pd.DataFrame(data)
8  df.set_index('Player', inplace=True)
```

1. Selecting a single row by index.

```
df.iloc[0]
```

0th index is belongs to 'Alice',

```
10  print(df.iloc[0])
```

PROBLEMS OUTPUT TERMINAL ... Python

```
PS C:\Users\ranga\Desktop\python> python main.py
Score    85
Match    10
Name: Alice, dtype: int64
PS C:\Users\ranga\Desktop\python> 
```

2. Selecting multiple rows by indices.

Provide the list of row indices needs to be accessed.

```
df.iloc[[0,2]]
```

0th and 2nd indices are belongs to 'Alice' and 'Charlie'

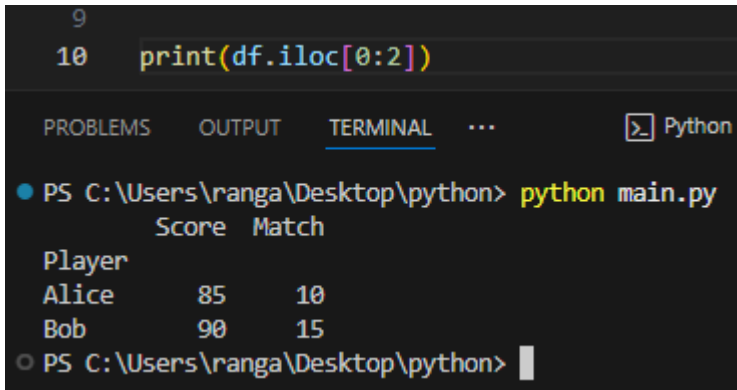
```
10  print(df.iloc[[0, 2]])
```

PROBLEMS OUTPUT TERMINAL ... Python

```
PS C:\Users\ranga\Desktop\python> python main.py
      Score  Match
Player
Alice      85     10
Charlie     95     20
PS C:\Users\ranga\Desktop\python> 
```

3. Selecting a range of rows by indices

```
df.iloc[0:2]
```



```
9
10 print(df.iloc[0:2])
```

PROBLEMS OUTPUT TERMINAL ... Python

PS C:\Users\ranga\Desktop\python> python main.py

| | Score | Match |
|--------|-------|-------|
| Player | | |
| Alice | 85 | 10 |
| Bob | 90 | 15 |

PS C:\Users\ranga\Desktop\python>

4. Selecting specific rows and columns by indices

```
df.iloc[[0, 2], [1]]
```

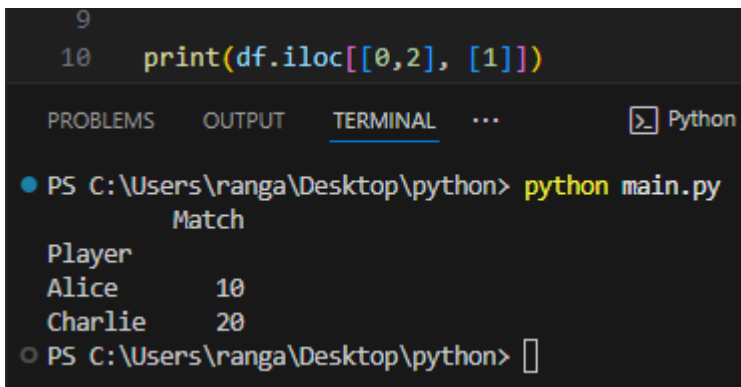
Ex:

0th row index : 'Alice'

2nd row index: 'Charlie'

1st column index: 'Match' (not the 'Scores' because 1st Series is used as the index with `inplace=True`. So the 0th column index is 'Scores' and the 1st column index is 'Match')

This will select Alice and Charlie's Match counts.



```
9
10 print(df.iloc[[0,2], [1]])
```

PROBLEMS OUTPUT TERMINAL ... Python

PS C:\Users\ranga\Desktop\python> python main.py

| | Match |
|---------|-------|
| Player | |
| Alice | 10 |
| Charlie | 20 |

PS C:\Users\ranga\Desktop\python>

5. Selecting all rows and specific columns

```
df.iloc[:, [1]]
```

```
9
10 print(df.iloc[:, [1]])
```

PROBLEMS OUTPUT TERMINAL ... Python

PS C:\Users\ranga\Desktop\python> python main.py

| Match | |
|---------|----|
| Player | |
| Alice | 10 |
| Bob | 15 |
| Charlie | 20 |

PS C:\Users\ranga\Desktop\python> |

Important

Inclusive vs. Exclusive

When selecting a range,

`loc` is inclusive of both endpoints.

`iloc` is exclusive of the top endpoint.

Ex:

`df.loc['Alice', 'Charlie']` includes 'Charlie'

`df.iloc[0:2]` includes rows at indices 0 and 1 but not 2.